Implementation of a Coded Modulation for Deep Space Optical Communications

Michael K. Cheng, Bruce E. Moision, Jon Hamkins, and Michael A. Nakashima,
Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109-8099

***Abstract*— We present an efficient implementation of a coded modulation for the deep space optical channel. NASA designed this so called serially concatenated pulse position modulation (SCPPM) code to provide an optical link that can operate within one dB signal energy of the Shannon capacity during a nominal mission condition from Mars. Here, we describe some of the challenges in realizing the SCPPM decoder on a field-programmable gate array (FPGA). Through various architectural optimizations, we achieve a 6 Mbps decoder on a single FPGA. Moreover, we demonstrate that it is possible to communicate reliably on an efficient bits-per-photon count in an end-to-end SCPPM coded system.**[1]

I. Introduction

Communication over deep-space is difficult. Communications beams spread as the square of the distance between the transmitter and the receiver. For example, geosynchronous Earth orbit (GEO) satellites are about 40,000 kilometers (km) in altitude and the average Mars-Earth distance is 80 million km. Therefore, the extra distance that a communication beam would have to travel from Mars to Earth would make data transfer 4 million times more difficult than from a GEO satellite to Earth. The signal power required to meet this extra effort and to cover this distance squared loss is greater than 66 dBs!

One way to increase the transmission rate from deep-space is through the use of more powerful transmit and receive antennas. However, this comes at a cost in increased antenna sizes which makes realization impractical. Another way is to communicate using frequencies much higher than radio frequency (RF) such as that of optical signals. Beams at higher frequency are more directionally concentrated and this allows a more efficient reception of the transmit energy [1, Ch. 1].

NASA's legacy error-correcting code (ECC) design for RF communication is the concatenation of an inner convolutional code and an outer Reed-Solomon (RS) code [2]. Decoding is performed in one pass utilizing hard bit-decisions. The discovery of turbo codes [3] and their suboptimal but effective low-complexity iterative decoding provided NASA a new code family with improved coding gains. NASA's first use of turbo codes is on the Messenger spacecraft launched in August of 2004.

An efficient ECC design for the deep space optical channel is the serial concatenation of an inner high-order modulation code and an outer convolutional code, namely serially concatenated pulse-position modulation or SCPPM. Moision and Hamkins showed [4] that SCPPM has the best performance and complexity tradeoff when compared to RS-PPM and Low-Density Parity Check (LDPC)-PPM. We may approximate true ML decoding while limiting the SCPPM decoder complexity by iteratively decoding the modulation and the ECC. This is in fact the "turbo" principle and more details can be found in [5].

This paper is organized as follows: in Section II we provide a model of the optical communications channel. In Section III, we give an overview of the SCPPM code and its decoding algorithm. In Section IV, we discuss some of the challenges associated with hardware implementation of the SCPPM decoder and describe our efficient approaches in detail. In Section V, we present the performance of a stand-alone decoder and an end-to-end optical communications system employing SCPPM.

II. System Description

We consider an optical communications system that uses direct photon detection with a high-order pulse-position modulation (PPM) [1, Ch. 1.2]. An M-order PPM modulation uses a time interval that is divided into M possible pulse locations, but only a single pulse is placed into one of the possible positions. The position of the pulse is determined by the information to be transmitted. A diagram of the optical communications system in discussion is shown in Fig. 1. The information bits $\mathbf{u} = (u_1, u_2, \cdots, u_k)$ are independent identically distributed (i.i.d.) binary random variables assumed to take on the values 0 and 1 with equal probability. The vector $\mathbf{u}$ is encoded to $\mathbf{c} = (c_1, c_2, \cdots, c_n)$, a vector of n PPM symbols. At the receiver, light is focused on a detector that responds to individual photons as illustrated in Fig. 2. For each photon sensed, the detector produces a band-limited waveform for input to the demodulator. This waveform is used to estimate the photon count, k_i, within each slot i. On the Poisson channel, a nonsignaling slot has average photon count n_b and a signaling slot has average count $n_s + n_b$ so that the likelihood ratio of slot i is given by

$$LR(k_i) = e^{-n_s}\left(1 + \frac{n_s}{n_b}\right)^{k_i}. \tag{1}$$

More on the receiver design can be found in [6].

III. The Serially Concatenated Pulse-Position Modulation (SCPPM) code

The SCPPM encoder, shown in Fig. 3, consists of an outer $\left(3, \frac{1}{2}\right)$ convolutional code, a polynomial interleaver, and an inner accumulate PPM (APPM) code. The trellis that describes the inner code consists of 2 states and $M/2$ parallel branches between connecting states.

[1]The research described in this paper was carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

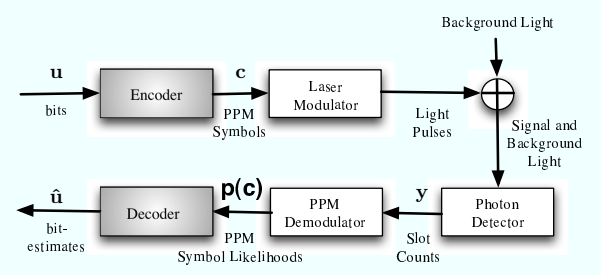

Fig. 1. An optical communication system.

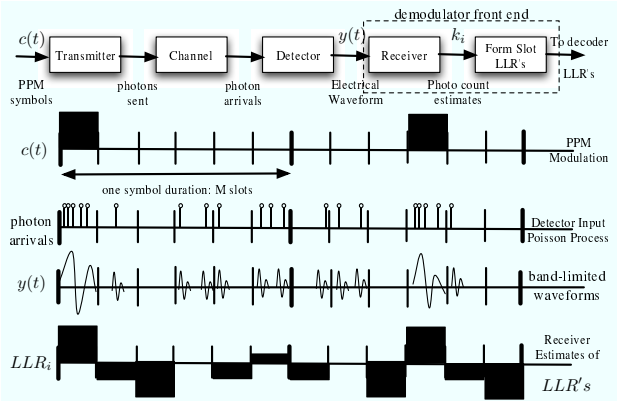

Fig. 2. From PPM symbols to decoder inputs.

A high level block diagram of the SCPPM decoder is illustrated in Fig. 4. The symbol I indicates input to the constituent decoders and O indicates output. The inner decoder operates on the APPM code and the outer decoder operates on the convolutional code. For each code trellis, the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [7] is used to compute the a-posteriori log-likelihood ratios (LLRs) from a-priori LLRs by traversing the trellis in forward and backward directions. Extrinsic information (the difference between the a-posteriori and a-priori LLRs) is exchanged in iteration rather than the a-posteriori LLRs to reduce undesired feedback. More on the SCPPM code and its decoding algorithm can be found in [4].

IV. Hardware Implementation

SCPPM decoding uses the turbo principle. However, due to the unique structure of SCPPM (for example, parallel edge transitions between stages in its trellis description), a straightforward application of classical turbo decoding techniques is inefficient. In this section, we first discuss the turbo-like part of SCPPM decoding and then present novel optimizations that led to a fast FPGA decoder implementation.

A. The Turbo-Like Part

Each constituent decoder applies the BCJR algorithm to the trellis that describes the corresponding code. Operations are performed in the log-domain to avoid multiplications which are costly to implement in hardware. This approach is known as log maximum a-posteriori (log-MAP) decoding [8]. Each log sum of exponentials can be expressed as the max of the exponents plus an adjustment term. This operation is known as the maxstar function:

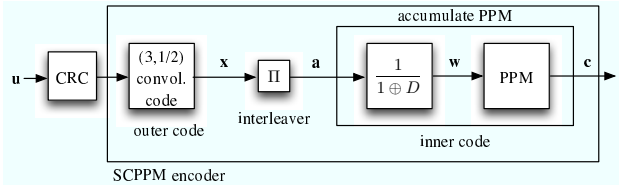

Fig. 3. The SCPPM encoder.

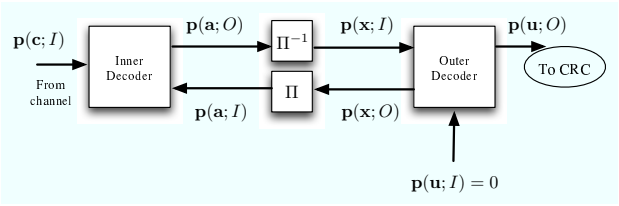

Fig. 4. The SCPPM decoder. The interleaver is labeled Π.

$$\overset{*}{\max}\ (x,y) \triangleq \ln\left(e^x + e^y\right) = \max(x,y) + \ln\left(1 + e^{-|x-y|}\right). \tag{2}$$

The adjustment term can be precomputed and stored in a lookup table to reduce complexity at an increase in memory usage. We can also ignore the adjustment term entirely to save on memory – this approach is known as max log-MAP decoding. Some of the loss incurred from this approximation can be recovered by scaling the extrinsic information that is passed between the inner and outer decoder [9].

B. Novel Optimizations

1) Parallel Trellis Edges: The trellis that describes the inner accumulate-PPM (APPM) code contains many parallel edges as seen in Fig. 5. To efficiently handle this large number of parallel edges, Barsoum and Moision [4] use the fact that the maxstar operation distributes over additions and developed a method of grouping the many trellis edge calculations per stage into one, and this combined value can be computed in a pipeline. We use notations that are standard in description of the BCJR algorithm. An edge e connects an initial state $i(e)$ with a terminal state $t(e)$. The backward recursion log-domain state metric β for state s and stage k is computed as:

$$\beta_k(s) = \overset{*}{\max}_{\tilde{s}\in\{s,\tilde{s}\}}\{\beta_{k+1}(\tilde{s}) + \gamma'_{k+1}(s,\tilde{s})\}. \tag{3}$$

The log-domain edge metrics are calculated as

$$\gamma'_k(s,\tilde{s}) = \overset{*}{\max}_{e:i(e)=s,t(e)=\tilde{s}}\{\gamma_k(e)\}. \tag{4}$$

Since the γ'_ks, or we refer to as "Super Gammas", are not a function of a recursively computed quantity, they may be precomputed via a pipeline and this reduces the edge computation time per trellis stage to one clock cycle. The α's are formed similarly.

2) Partial Statistics: To reduce the channel likelihood storage requirements, we may discard the majority of the channel likelihoods and use partial statistics [10]. This may be accomplished by processing only a subset consisting of the largest slot likelihoods during each symbol duration–the likelihoods corresponding to the slots with the largest number of observed photons. The observation of the remaining slots

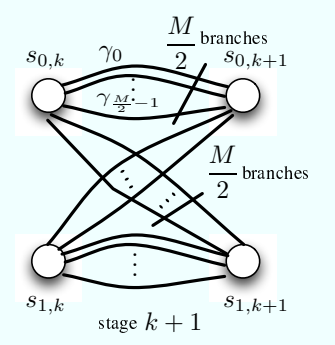

Fig. 5. A stage of the inner accumulate PPM (APPM) trellis.

is set to the mean of a noise slot. In low background noise, a small subset may be chosen with negligible loss.

3) Interleaver Design: The interleaver in Fig. 4 is characterized by a second order polynomial $f(j) = aj + bj^2$. The bit position j is mapped to the position $[f(j)]_N$ where $[\cdot]_N$ is the mod N operation. Let us factor the codeword length N as products of primes, that is, $N = p_1^{j_1} p_2^{j_2} \cdots p_\ell^{j_\ell}$. Any polynomial with $b = p_1 p_2 \cdots p_\ell$ and a set to a number that does not have p_1,p_2,$\cdots$, or p_ℓ as a factor is a candidate interleaver [11]. The mapping for the $(j+i)$th interleaver position can be expressed as a function of the current interleaver position j:

$$[f(j+i)]_N = [f(j) + g(i,j)]_N \tag{5}$$

where $g(i,j) = 2ijb + i(a+bi)$. This property enables an algorithmic implementation that does not require the mapping to be precomputed and stored [12]. For a codeword length of $N = 15120$ bits, we found the polynomial $f(j) = 11j + 210j^2$ to have good performance.

For an M-order PPM modulation, the inner decoder processes a PPM symbol (or $\log_2 M$ bit LLRs) per trellis stage. A straightforward scheduling would be to read one LLR from the interleaver memory per clock. This approach incurs a long latency because the inner decoder would have to wait $\log_2 M$ clocks before proceeding to the next stage. To design an interleaver that allows one clock read/write access we partition the interleaver memory into $\log_2 M$ modules. We illustrate our idea using $M = 64$. For these parameters, the interleaver is divided into six modules as shown in Fig. 6. Each outer trellis stage decoding produces two LLRs and these are written in permuted order, with mapping given by (5) and computed on-the-fly, to two of the six memory modules. Each inner trellis stage decoding requires six LLRs from the interleaver. These LLRs are obtained by reading the same entry out of each of the six modules in parallel, that is, first the top entries out of the six modules, then the second entries, third entries, and so on. Thus, a stage of LLRs are stored and fetched from the interleaver in one clock. The deinterleaver is designed similarly.

4) Decoder Windowing: The inner trellis consists of $N/\log_2 M$ symbols or stages. The outer code trellis is a rate 1/2 code and has $N/2$ stages. For PPM orders M greater than 4, the outer code trellis will contain more stages than the

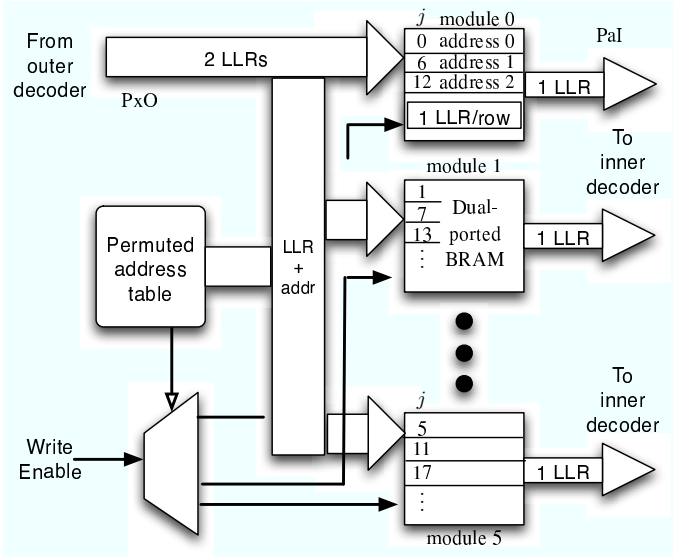

Fig. 6. A one clock access interleaver design. The permuted address table entries can be computed on-the-fly.

inner code trellis. If a straightforward scheduling is used to traverse the two trellises, the inner decoder will have to wait longer for the outer decoder to complete a trellis pass. It is advantageous to have both decoders complete an iteration in the same amount of time. The latency in this case is reduced because the wait time of the inner decoder is reduced. To do so, we partition the outer code trellis into distinct windows and decode the windows in parallel.

For $M = 64$ and $N = 15120$, the outer decoder is windowed by three. In this scenario, we observed through simulations that no warmup windows are required to obtain a performance close to that of the non-windowed decoder. While the concept of decoder windowing is not new, we did not find in literature a cyclic redundancy check (CRC) implementation that works efficiently with a windowed decoder.

5) Cyclic Redundancy Check: A CRC can be used together with iterative turbo decoding to stop iterations and flag codeword errors. In windowed-based turbo decoding, the bits to be input to the CRC are generated in parallel more than one at a time. Therefore, the conventional serial input linear feedback shift register (LFSR) circuit that implements a CRC needs to be modified to handle this parallelism.

Let us write a length k binary message block $\mathbf{m} = (m_{k-1}, m_{k-2}, \cdots, m_0)$, that is to be protected by a CRC, in polynomial form:

$$m(x) = m_{k-1}x^{k-1} + m_{k-2}x^{k-2} + \cdots + m_0 \tag{6}$$

Let the length n CRC protected codeword be $\mathbf{c} = (c_{n-1}, c_{n-2}, \cdots, c_0)$ or

$$c(x) = c_{n-1}x^{n-1} + c_{n-2}x^{n-2} + \cdots + c_0 \tag{7}$$

and the CRC generator be

$$g(x) = g_{n-k}x^{n-k} + \cdots + g_0. \tag{8}$$

The CRC polynomial $r(x)$ is calculated by first shifting the message polynomial left by $n-k$ positions and then by taking

the modulo $g(x)$ operation

$$r(x) = R_{g(x)}\left[m(x) \cdot x^{n-k}\right], \tag{9}$$

where $\deg[r(x)] < n-k$. The codeword polynomial is expressed as $c(x) = m(x) \cdot x^{n-k} + r(x)$.

To verify the CRC of a codeword block $\hat{c}(x) = c(x) + e(x)$ that might be corrupted by an error polynomial $e(x)$, we compute $R_{g(x)}[\hat{c}(x)] = R_{g(x)}[e(x)]$. Therefore, if the remainder is zero, the CRC passes and the error polynomial is zero. If the remainder is nonzero, then the codeword is corrupted. Note that we won't be able to construct the error polynomial $e(x)$ from the CRC remainder $R_{g(x)}[e(x)]$.

In windowed-based turbo decoding, the output bit streams to be fed into the CRC are generated in parallel. We describe how a CRC circuit can be modified to handle this parallelism.

Let the code trellis be partitioned into j distinct windows. The codeword polynomial can be written as

$$c(x) = c_1(x)\, x^{s_1} + c_2(x)\, x^{s_2} + \cdots + c_j(x). \tag{10}$$

We can then write the check polynomial as

$$\begin{aligned} R_{g(x)}[c(x)] &= R_{g(x)}[c_1(x)\, x^{s_1} + c_2(x)\, x^{s_2} + \cdots + c_j(x)] \\ &= R_{g(x)}\left[R_{g(x)}[c_1(x)\, x^{s_1}] + R_{g(x)}[c_2(x)\, x^{s_2}]\right. \\ &\quad \left. + \cdots + R_{g(x)}[c_j(x)]\right] \\ &= R_{g(x)}\left[R_{g(x)}[c_1(x)\,\kappa_1(x)]\right. \\ &\quad \left. + R_{g(x)}[c_2(x)\,\kappa_2(x)] + \cdots + R_{g(x)}[c_j(x)]\right], \end{aligned} \tag{11}$$

where $\kappa_i = R_{g(x)}[x^{s_i}]$, $i = 1, 2, \cdots, j-1$, and each $\kappa_i(x)$ can be pre-calculated. The CRC LFSR circuit for the window-based decoder will consist of both feed-forward and feedback tap connections. The feed-forward taps are given by the XOR of $\kappa_i(x)$'s and the feedback taps are given by the generator $g(x)$. A generic LFSR circuit that multiplies an arbitrary polynomial $\kappa_i(x)$ and divides an arbitrary polynomial $g(x)$ can be found in [13].

V. System Performance

A. Stand-Alone Decoder

The SCPPM decoder for $M = 64$ and $N = 15120$ is currently implemented on a Xilinx Virtex II-8000 FPGA part, speed grade 4 (XC2V8000-4), which sits on a Nallatech BenDATA-WS board. The memory requirement is reduced by taking only the top 8 channel LLRs as decoder input. The LLR input quantization is 8 bits, 5 for dynamic range and 3 for decimal precision. We have implemented two versions of the decoder: The first is the log-MAP decoder with clipping and normalization circuits. The second is the max log-MAP decoder with modulo arithmetic (which allows operations to overflow without the need for normalization) and windowing. The outer code trellis is windowed by three.

The total FPGA resource utilization (slices for logic and BRAM for memory) for the two decoders are given in Table I. Note that the max log MAP decoder is a 4 million gate implementation (53% of 8 million total gates). Windowing of the outer trellis by three also led to three times the BRAM consumption in the outer decoder. Miscellaneous numbers include blocks that consume resources such as the circuitries and memories instantiated for the interleaver, deinterleaver, and FPGA interface. The $\overset{*}{\max}$ lookup tables (LUTs) for the log-MAP decoder are realized as read-only memories (ROMs).

log-MAP	Used/Total	Utilization	Inner	Outer	Misc.
BRAM	101/168	60%	19%	9%	32%
Slices	30174/46592	64%	52%	6%	6%

max log-MAP	Used/Total	Utilization	Inner	Outer	Misc.
BRAM	158/168	89%	19%	30%	40%
Slices	24587/46592	53%	32%	15%	6%

TABLE I
SCPPM DECODERS ON THE VIRTEX-II 8000 FPGA.

The max log-MAP decoder, with all of the proposed architectural optimizations, supports a maximum clock rate of 60 MHz and a throughput of 6.4 Mbps based on 7 average iterations .

The decoder performance is shown in Fig. 7. The word error rate (WER) is plotted versus n_s, the average number of signal photons per PPM pulse slot in dB. The average noise photons per slot is $n_b = 0.2$. Each codeword consists of 7560 information bits. A word error is declared when the decoder decision could not converge to the correct codeword in the maximum number of allowed iterations which is set at 32. Out of the 7560 bits, 2 bits are used to terminate the trellis and 22 bits are used for Cyclic Redundancy Check (CRC). The CRC polynomial is $x^{22} + x^5 + x^4 + x^3 + 1$ and has an undetected word error probability of approximately $7 \cdot 2^{-22} = 1.67 \times 10^{-6}$ assuming 7 average iterations. To reduce the undetected rate, the decoder runs a minimum number of iterations first before validating the CRC.

We make the following observations in the performance plot. Fixed-point implementation (circle-line) has a 0.1 dB loss compared to the floating-point decoder (dashed-line). Clipping and and normalization of the state metrics led to a floor at 10^{-5}. Max log-MAP decoder with fast modulo normalization (square-line) has a 0.6 dB loss compared to log-MAP decoding (circle-line). Max log-MAP decoder with a scaling of the extrinsic information by 0.5 (diamond-line) recovers 0.4 dB out of the 0.6 dB lost. Although not shown here, the SCPPM code scheme has approximately a 3 dB signal energy gain over an equivalent rate RS-PPM code scheme.

B. End-to-End System

We have successfully demonstrated [14] an end-to-end SCPPM optical communications system as shown in Fig. 8. We are able to deliver quality MPEG-2 video from a camera to a display using this setup. The transmitter employs a 1064 nm wavelength (Nd: YAG) solid state laser to modulate a stream of SCPPM encoded symbols. The PPM pulses are then sent over a fiber optic channel. At the receiving end, a Hybrid Photo-Multiplier Tube (HPMT) photon counting detector is used and the receiver assembly converts the photon counts into LLRs for our FPGA decoder. The results of the experimental runs at various operating points are plotted in Fig. 9. There

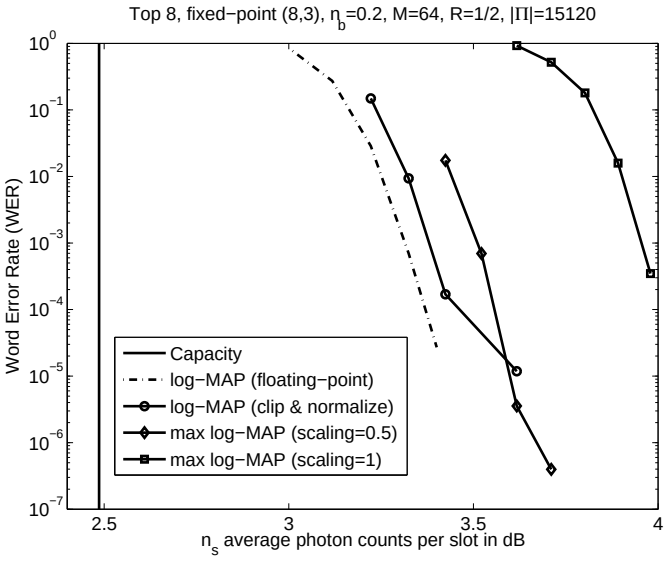

Fig. 7. SCPPM decoder performance on the Poisson channel.

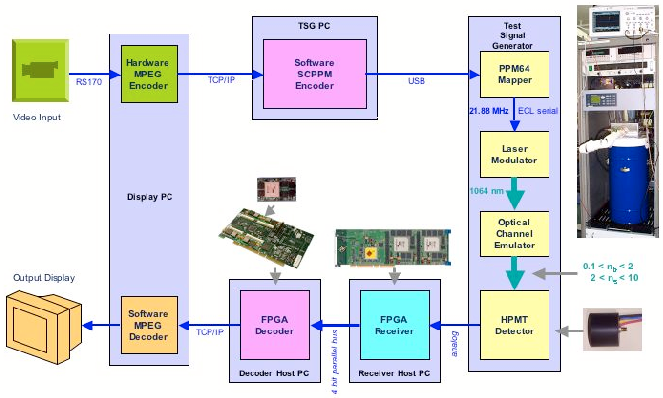

Fig. 8. An end-to-end SCPPM optical link demonstration.

are two experimental runs, one at 4 Mbps and the other at 6 Mbps both use only the top 8 statistics and a maximum of 7 iterations. These two curves are compared to a curve generated by using a software simulated Poisson channel and the stand-alone FPGA decoder. We see that the experimental curves match very closely to the stand-alone FPGA result. The end-to-end performs within 1.5 dB of channel capacity. At a frame loss rate of 10^{-5} the number of signal photons per pulse slot is 2.67 and this corresponds to 3/2.67=1.12 information bits per photon.

VI. Summary

NASA designed SCPPM as a capacity approaching coded modulation for deep space optical communications. Due to the code structure, direct application of turbo decoding is inefficient. In this work, we presented various implementation techniques that produced a fast hardware decoder. Potential coded modulation design and realization for future optical links can benefit from our techniques. Incorporating our hardware architecture, we developed a 6 Mbps stand-alone SCPPM decoder on an FPGA. We also demonstrated an end-to-end SCPPM optical communications system that performs within

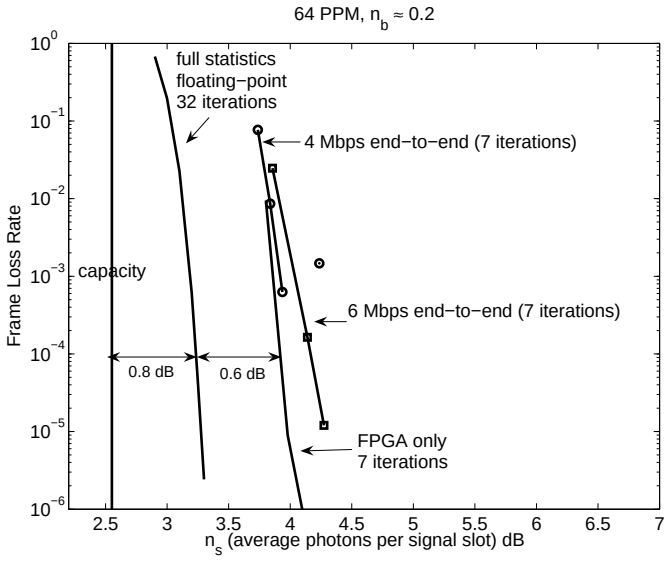

Fig. 9. End-to-end system performance. A frame is equivalent to a codeword of 7560 information bits.

1.5 dB of capacity and delivers 1.12 information bits per photon at a desirable decoder error rate.

References

[1] H. Hemmati, *Deep Space Optical Communications*. John Wiley & Sons Inc., 2006. ISBN 0-470-04002-5.

[2] CCSDS, "Telemetry channel coding." Consultative Committee for Space Data Systems Standard 101.0-B-6, Oct. 2002.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.*, vol. 2, (Geneva, Switzerland), pp. 1064–1070, IEEE, May 1993.

[4] B. Moision and J. Hamkins, "Coded modulation for the deep space optical channel: serially concatenated PPM," *JPL Interplanetary Network Progress Report*, vol. 42-161T, May 2005.

[5] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A soft-input soft-output maximum a posteriori (MAP) module to decode parallel and serial concatenated codes," *The Telecom. and Data Acquisition Progr. Rep.*, vol. 42, pp. 1–20, Nov. 1996.

[6] K. J. Quirk and L. B. Milstein, "Optical PPM with sample decision photon counting," in *Proc. IEEE Global Telecom. Conf.*, (St Louis, MO, USA), IEEE, Dec. 2005.

[7] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284–287, March 1974.

[8] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Select. Areas Commun.*, vol. 16, pp. 260–264, Feb. 1998.

[9] P. H. Wu and S. M. Pisuk, "Implementation of a low complexity, low power, integer-based turbo decoder," in *Proc. IEEE Global Telecom. Conf.*, vol. 2, (San Antonio, Texas), pp. 946–951, IEEE, 2001.

[10] B. Moision and J. Hamkins, "Reduced complexity decoding of coded-pulse modulation using partial statistics," *JPL Interplanetary Network Progress Report*, vol. 42-161, May 2005.

[11] O. Y. Takeshita, "On maximum contention-free interleavers and permutation polynomials over integer rings," *IEEE Trans. Inform. Theory*, vol. 52, pp. 1249–1253, March 2006.

[12] M. K. Cheng, B. E. Moision, J. Hamkins, and M. A. Nakashima, "An interleaver implementation for the serially concatenated pulse position modulation decoder," in *IEEE Proc. Int. Symp. Circuits and Systems*, (Kos, Greece), May 2006.

[13] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes*. Cambridge, MA, USA: The M.I.T. Press, 1961.

[14] A. Biswas and et. al., "Palomar receive terminal for the Mars Laser Communications Demonstration project," *IEEE Special Issue on Tech. Adv. in Deep Space Comm. and Tracking*, Nov. 2006. Submitted.